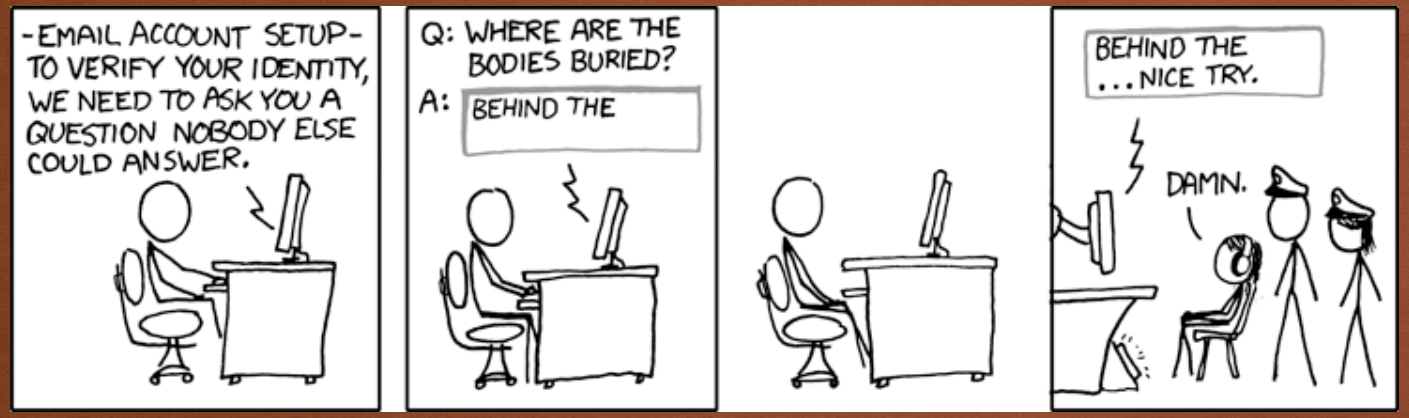
Operating Systems
Part 3



https://xkcd.com/565

Roles of an OS

1. User Interface Management (a receptionist)
2. Program Scheduling and Activation (a dispatcher)
3. Efficient Resource Allocation (an efficiency expert)
4. Deadlock Detection and Error Detection (a traffic officer)
5. **Control of access to the system and data (a security guard)**

Operating System as a Security Guard

- Time-shared OSs created another problem : Security
  - How do we prevent inexperienced or malicious users from inadvertently creating problems for other users of the computer?
  - This is also one of the jobs of the OS.

- There needs to be some mechanism to control access to the system and data.
  - Authentication (Security)
    - prove who we are to the OS.
  - Authorization (Protection)
    - Once we have proven who we are we can do whatever we are allowed to do.
    - Usually programs we run execute with our privileges

Authentication

How can we **authenticate** (prove who we are to the OS)?

- Username/Passwords
- Biometric - finger print, face, retina scan, voice
- Physical key or card (maybe a smart card)
- The computer sends a code to your phone or email
- 2FA - two factor authentication, combining two of the above

- Which of the above is the worst? Which is the most common?

Authorization

- The OS needs to protect **objects** from illegal access by **subjects**.
- Objects include
  - files, memory, locks, network connections, programs, devices
- Subjects include
  - people, processes (programs running)
- There are two main ways for the OS to keep track of who is allowed to do what.
  - Keep a list associated with each **subject** (a capability list)
    - entries like - can read from file X
  - Keep a list associated with each **object** (an access control list)
    - entries like - A can read

CAPABILITIES AND ACLS

- The list of information associated with each user (or process) is called a **capability list.**
  - Users do not have the capability to change their capability lists.
- The list of information associated with each object is called an **access control list (ACL).**
- Every time an object is to be used the OS should check to see if the requested access is allowed.
- If the OS only checks once for each object we can have the TOCTOU problem - time of check to time of use.

Linux File Permissions

Linux provides a simple approach to file permissions.

Ownership

Each file/directory has an owner and a group associated with it.

Owner : person who created the file and can change its privileges.

Group : a collection of users

At a time, a file can belong to only one user and one group.

Permissions

A file/directory also permissions associated with it.

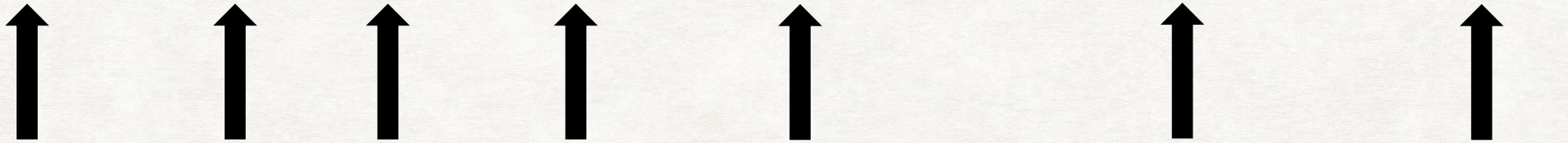
They help determine which users may read (**`r`**), write (**`w`**) or execute (**`x`**) the file.

File Owner and Permissions

When you type :

```
$ ls -l /usr/bin/top
```

```
-r-xr-xr-x  1    root     root     109064   Nov 22 2016 07:18 /usr/bin/top
```



File Permissions | Link Count | Owner | Group | File Size | Modification Date/Time | File Name

```
$ ls -l
```

```
-rwx------ 1 asha946 all 13206 Jan 10 11:20 lecture1
-rwx------ 1 asha946 all 24569 Jan 10 11:23 lecture2
-rwx------ 1 asha946 all 19147 Jan 10 14:50 lecture3
drwx------ 2 asha946 all  2048 Jan 10 11:36 slides
```

File Permissions | Owner | Group

Permissions of a file

```
$ ls -l
-rw- r-- r-- 1 asha946 all 13206 Jan 10 11:20 lecture1
-rwx r-x --- 1 asha946 all 24569 Jan 10 11:23 program1
```

For file `lecture1`:

- owner of the file (`asha946`) has read & write permission
- group (`all`) members have read permission
- others have read permission

For file `program1`:

- owner (`asha946`) has read, write & execute permissions
- group (`all`) members have read & execute permissions
- others have no permissions at all (cannot read, write or execute)

Changing the owner/group of a file

```
$ chown <owner>[:GROUP] <filename>
```

<owner> – name of the new owner (or ID).

Example,

```
chown asha946 file1
chown asha946:group1 file1
```

This command requires the user to have root (or sudo) privileges.

```
$ chgrp <group> <filename>
```

<group> – name of the new group or

groupid (must be prefixed with the + symbol)

The owner of a file may change the group of the file to any group of which that owner is a member.

The root may change the group arbitrarily.

Changing the permissions of a file

```
chmod options permissions <filename>
```

There are **two** ways to set permissions when using the chmod command:

1. Symbolic mode:

```
$ ls -l
-rwxr-x--- 1 asha946 all 24569 Jan 10 11:23 program1
                                    U   G   O
$ chmod g+w  program1          ==>  -rwxrwx---
$ chmod o+x  program1          ==>  -rwxrwx--x
$ chmod ug-x program1          ==>  -rw-rw---x
u=user, g=group, o=other (world)
```

Changing the permissions of a file

```
chmod options permissions <filename>
```

There are **two** ways to set permissions when using the `chmod` command:

2. Absolute mode (uses octal values):

For each column, User, Group or Other we can set values from 0 to 7.

Octal value	0	1	2	3	4	5	6	7
Meaning	`---`	`--x`	`-w-`	`-wx`	`r--`	`r-x`	`rw-`	`rwx`

Changing the permissions of a file

```
chmod options permissions <filename>
```

There are **two** ways to set permissions when using the chmod command:

2. Absolute mode (uses octal values):

```
                                        U  G  O
$ chmod 770  program1          ==>   -rwxrwx---
$ chmod 751  program1          ==>   -rwxr-x--x
$ chmod 661  program1          ==>   -rw-rw---x
```

Exercise Time

1. What is the command to change the ownership of file `program1` from `asha946` to `ssy478`?

2. Can you issue this command if you are logged in as user `asha946`?

3. You are logged in as user `ssy478`.

   `ls -l` gives the following output.

   ```
   -rwxr----- 1 ssy478 all 24569 Jan 10 11:23 program1
   ```

4. Will you be able to execute the file `program1`?

5. Give your `group` the ability to execute this file.

Generations of an OS : Continued

Generation	Dates	Characteristics
First	1945 - 1955	**Naked Machines – no OS** Programmers operated the machine themselves
Second	1955 – 1965	**Batch Operating Systems** Several programs were grouped into a “batch” An operator loaded this batch of programs onto an input tape which was then fed to a main computer that executed them sequentially. Improved system utilization from the single user scenario.
Third	1965 – 1985	**Multiprogrammed OS** The OS keeps several programs in memory. If the currently running program pauses for I/O; the OS picks up one of the other “ready” process from a queue. Improves processor utilization by keeping the processor “busy” most of the time.
	1970s	**Time-sharing OS** Interactive use of a central computer system. Users sit at terminals (“dumb”) and are connected to the central computer via communication links
Fourth	1985 - Present	**Network Operating Systems** Client-server computing Remote access to resources
Fifth	Future OSs	?

Fourth generation: **Network Operating System (1985- Present)**

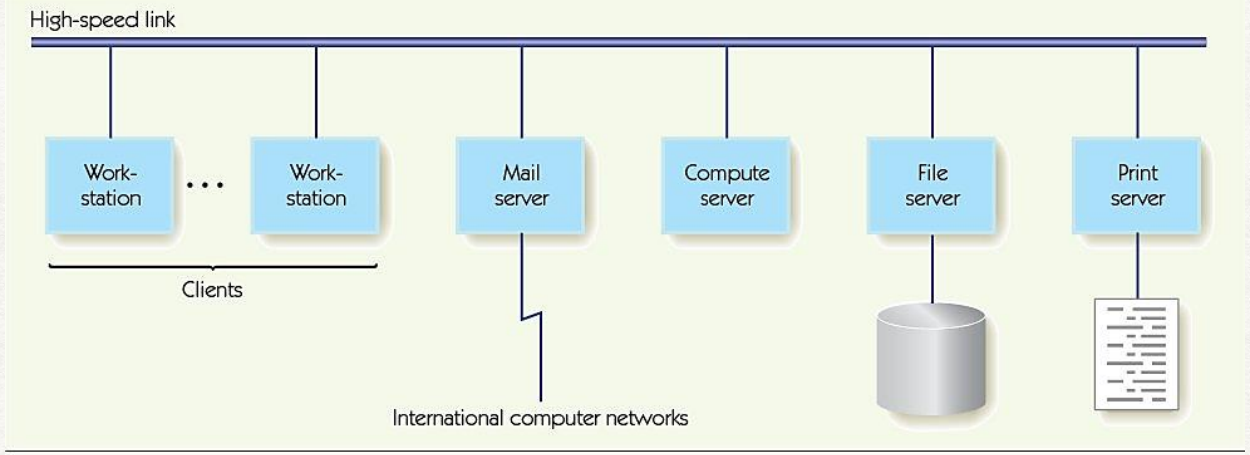
- The computing paradigm changed rapidly from the centralized environment of time-sharing systems to a *distributed environment*.
  - cheap **PCs (Personal Computers)**
  - Peripherals (laser printers, large disk drives, tape backup units and specialized software packages) were still expensive.

- A virtual environment was needed where users could have access to both **local computation** and **remote access** to shared resources.

- A Network Operating System provides access
  - Local resources of a computer
  - Resources on a Local Area Network (LAN).

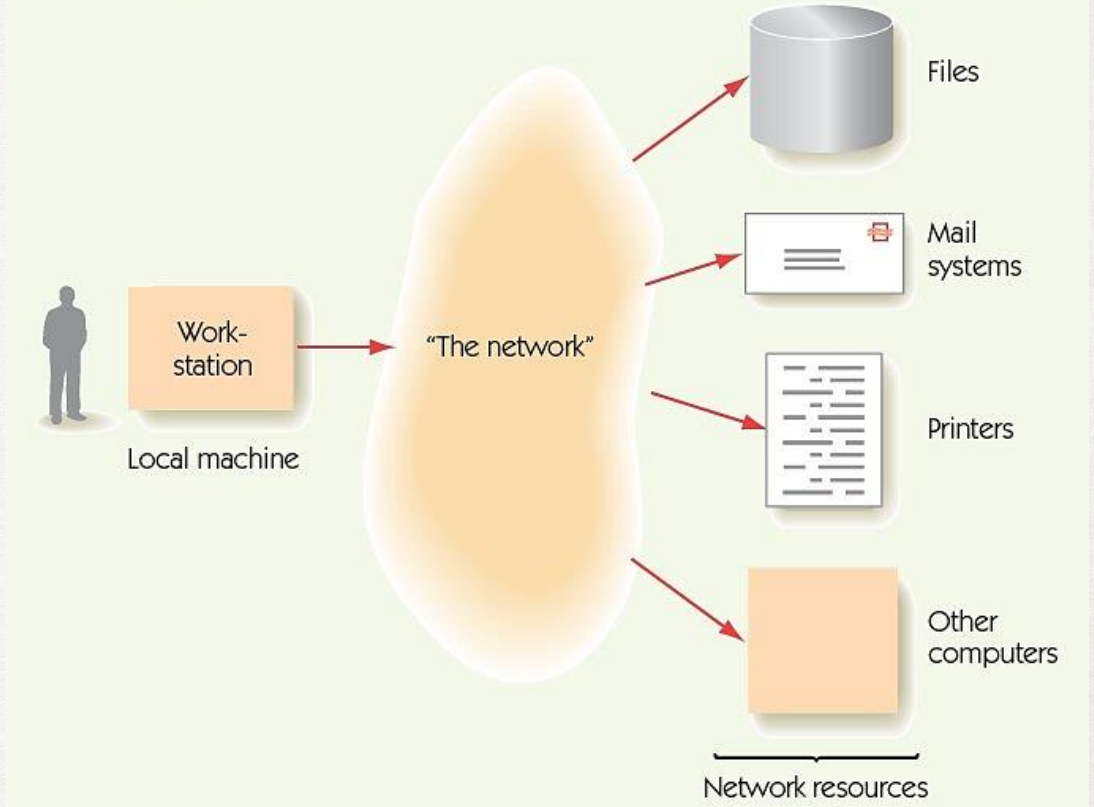
Network Operating Systems

FIGURE 6.20



A local area network

FIGURE 6.21



The virtual environment created by a network operating system

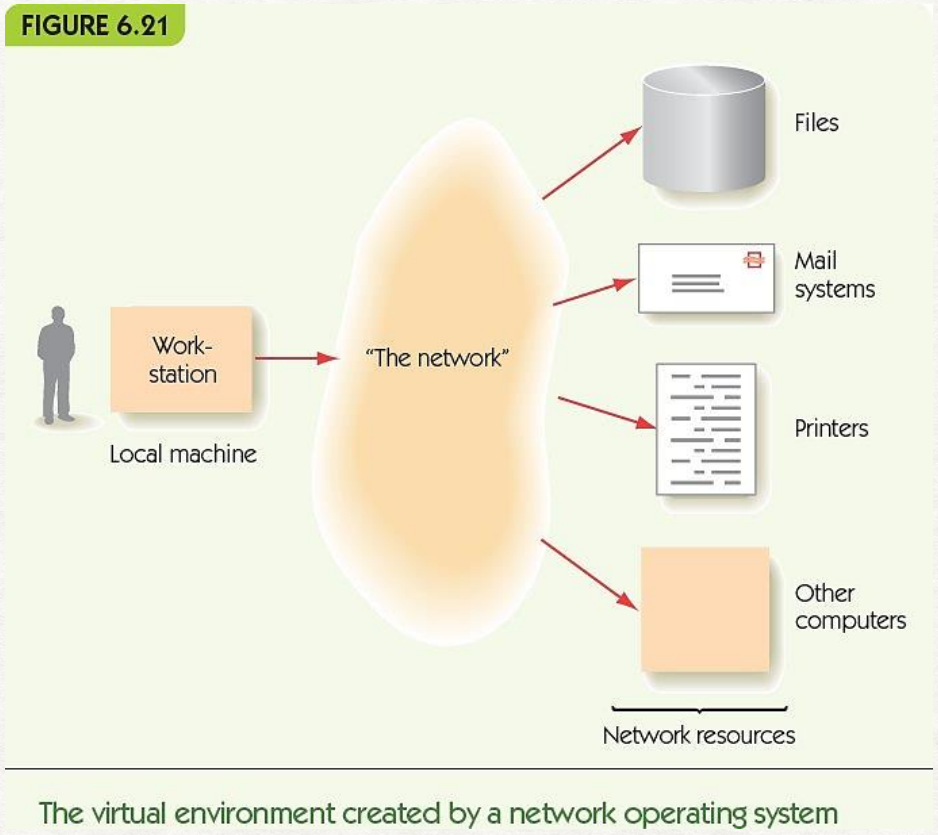
Fifth generation, the near future

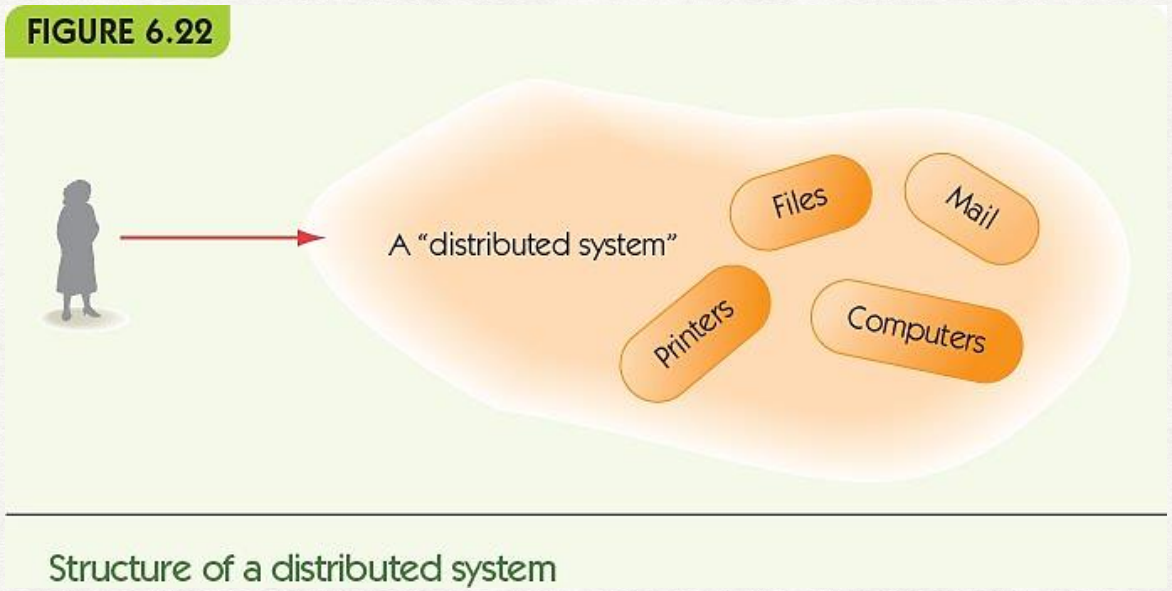
- **Distributed Operating Systems**
  - **A distributed operating system connects multiple independent computers via a network to perform tasks similar to a single computer.**
  - Users are unaware of the machines on which the resources are stored
  - Cloud Computing

- Multimedia interfaces (integrate images, speech, and video seamlessly)

- Parallel processing system to perform multimedia and to permit larger scale tasks

Network OSs vs Distributed OSs





Users are aware of the multiplicity of machines.

Users are unaware of the multiplicity of machines.

Fifth generation, the near future

- Distributed Operating Systems in which users don’t know where the resources are stored
  - Cloud computing
- Multimedia interfaces (integrate images, speech, and video seamlessly)
- Parallel processing system to perform multimedia and to permit larger scale tasks

Generations of an OS

Generation	Dates	Characteristics
First	1945 - 1955	**Naked Machines – no OS** Programmers operated the machine themselves
Second	1955 – 1965	**Batch Operating Systems** Several programs were grouped into a “batch” and fed to a main computer Improved system utilization from the single user scenario.
Third	1965 – 1985	**Multiprogrammed OS** The OS keeps several programs in memory. If the currently running program pauses for I/O; the OS picks up one of the other “ready” process from a queue. Improves processor utilization by keeping the processor “busy” most of the time.
	1970s	**Time-sharing OS** Interactive use of a central computer system. Users sit at terminals (“dumb”) and are connected to the central computer via communication links
Fourth	1985 - Present	**Network Operating Systems** Client-server computing Remote access to resources
Fifth	Present - Future	**Distributed computing environments** Multimedia UIs Massively Parallel OSs